

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ
по дисциплине «Разработка ПО информационных систем»
Тема: Статистика студенческих отчётов

Студенты гр. 6304

Рыбин А.С.

Ковынёв М.В.

Тимофеев А.А.

Преподаватель

Заславский М.М.

Санкт-Петербург

2019

ЗАДАНИЕ

Студенты: Рыбин А.С., Ковынёв М.В., Тимофеев А.А..

Группа 6304.

Тема проекта: Разработка приложения «Статистика студенческих отчётов».

Исходные данные: Необходимо сделать приложение, которое позволяет:
стемматизировать текст отчетов, исключать стоп слова, вычислять статистику
употребления отдельных слов, определять словарный запас, определять
пересечения словарных запасов.

Предполагаемый объем пояснительной записки:

Не менее 15 страниц (обязательны разделы «Содержание», «Введение»,
«Выводы», «Список использованных источников»).

Дата выдачи ИДЗ: __.__.____

Дата сдачи ИДЗ: __.__.____

Дата защиты ИДЗ: __.__.____

Студенты гр. 6304

Рыбин А.С.

Ковынёв М.В.

Тимофеев А.А.

Преподаватель

Заславский М.М.

АННОТАЦИЯ

Исследование возможностей **нереляционных** баз данных на примере популярного **NoSQL** решения *MongoDB*. Проектирование и разработка приложения, ориентированного на хранение и анализ данных используя NoSQL БД (статистика студенческих отчётов). Сравнительный анализ **NoSQL** с традиционным **SQL** решением по сложности и количеству запросов и требуемому объёму памяти.

SUMMARY

Explore the capabilities of **non-relational** databases on the example of the popular *MongoDB* **NoSQL** solution. Design and development of an application focused on storing and analyzing data using NoSQL DB (statistics of student reports). Comparative analysis of **NoSQL** with traditional **SQL** solution for the complexity and number of queries and the required amount of memory.

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ	6
1.1. Актуальность.....	6
1.2. Постановка задачи	6
1.3. Предлагаемое решение.....	6
2. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ	7
2.1. Текущие	7
2.2. Перспективные.....	7
3. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ	8
3.1. Макет UI	8
3.2. Сценарии использования для задачи	9
Импорт/экспорт данных	9
Импорт.....	9
Экспорт.....	9
Загрузка нового отчета в систему	9
Основной сценарий	9
Опциональные шаги.....	9
Альтернативный сценарий (ошибка)	9
Просмотр статистики.....	9
Основной сценарий	9
Опциональные шаги.....	10
4. МОДЕЛЬ ДАННЫХ	11
NoSQL модель данных (MongoDB).....	11
Подробное описание и расчёт объёма	11
Запросы	12
SQL модель данных	14
Подробное описание и расчёт объёма	14
Запросы	15
SQL vs NoSQL	16
5. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ	17
Экраны приложения	17
Использованные технологии	21

Ссылки:	21
ВЫВОДЫ	22
Список использованных источников	23
ПРИЛОЖЕНИЕ А	24

1. ВВЕДЕНИЕ

1.1. Актуальность

Сама по себе идея нереляционных баз данных не нова, а использование нереляционных хранилищ началось ещё во времена первых компьютеров. Нереляционные базы данных процветали во времена мэйнфреймов, а позднее, во времена доминирования реляционных СУБД, нашли применение в специализированных хранилищах, например, иерархических службах каталогов. Появление же нереляционных СУБД нового поколения произошло из-за необходимости создания параллельных распределённых систем для высокомасштабируемых интернет-приложений, таких как поисковые системы [1].

Поддержка гигантов индустрии менее чем за пять лет привела к широкому распространению технологий NoSQL (и подобных) для управления «большими данными», а к делу присоединились другие большие и маленькие компании, такие как: IBM, Facebook, Netflix, eBay, Hulu, Yahoo!, со своими проприетарными и открытыми решениями [1].

1.2. Постановка задачи

Необходимо сделать приложение, которое позволяет: стемматизировать текст отчетов, исключать стоп слова, вычислять статистику употребления отдельных слов, определять словарный запас, определять пересечения словарных запасов студентов.

1.3. Предлагаемое решение

В качестве решения выступает *веб* приложение, которое позволяет загружать отчёты, хранить отчёты в систематизированном виде и получать различную статистику, связанную с **текстом отчётов**, принадлежащих студенту, группе или факультету, или кафедре.

2. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

2.1. Текущие

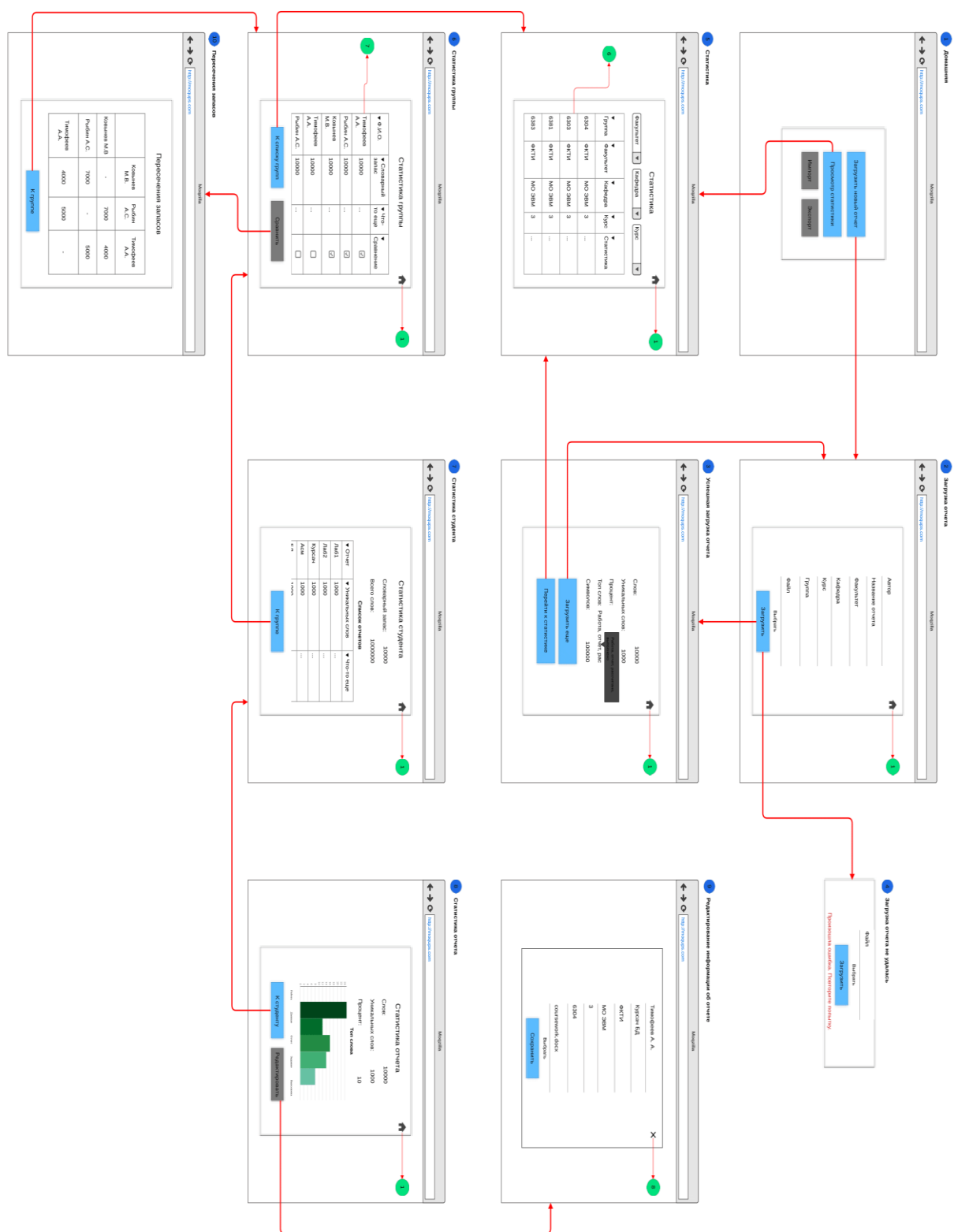
- Для хранения данных используется **MongoDB**
- Есть возможность массового *импорта/экспорта* из БД
- Есть возможность получить статистику по загруженному отчёту
- Есть возможность сравнить словарные запасы студентов

2.2. Перспективные

- Авторизация студентов и преподавателей
- Поддержка **doc, pdf, odt**

3. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

3.1. Макет UI



Макет также можно посмотреть по [ссылке](#).

3.2. Сценарии использования для задачи

Импорт/экспорт данных

Действующее лицо: **Пользователь**

Импорт

Пользователь на главной странице нажимает кнопку **“Импорт”**

Пользователь загружает файл в формате **json**

База данных отчетов обновляется в соответствии с загруженным файлом

Экспорт

Пользователь на главной странице нажимает кнопку **“Экспорт”**

Пользователю предлагается загрузить файл в формате **json**

Загрузка нового отчета в систему

Действующее лицо: **Пользователь**

Основной сценарий

Пользователь на главной странице нажимает кнопку **“Загрузить новый отчет”**

В форме ввода данных на следующей странице *пользователь* вводит мета информацию об загружаемом отчете и прикрепляет файл в формате **docx**

Пользователь нажимает кнопку **“Загрузить”**

Пользователь видит сообщение об удачной загрузке отчета и подробную статистику по загруженному отчету

Пользователь выполняет опциональные шаги

Опциональные шаги

- Для загрузки еще одного отчета *пользователь* нажимает кнопку **“Загрузить еще”**
- Для перехода к просмотру статистики *пользователь* нажимает кнопку **“Перейти к статистике”**

Альтернативный сценарий (ошибка)

- Пользователь видит сообщение об ошибке загрузки
- Пользователю предлагается повторить попытку загрузить отчет

Просмотр статистики

Действующее лицо: **Пользователь**

Основной сценарий

На главной странице *пользователь* нажимает кнопку **“Просмотр статистики”**

Пользователь с помощью селекторов: **“Кафедра”**, **“Курс”** и **“Факультет”** выбирает по каким группам отображать статистику

Пользователю отображается страница с агрегированной статистикой по группам

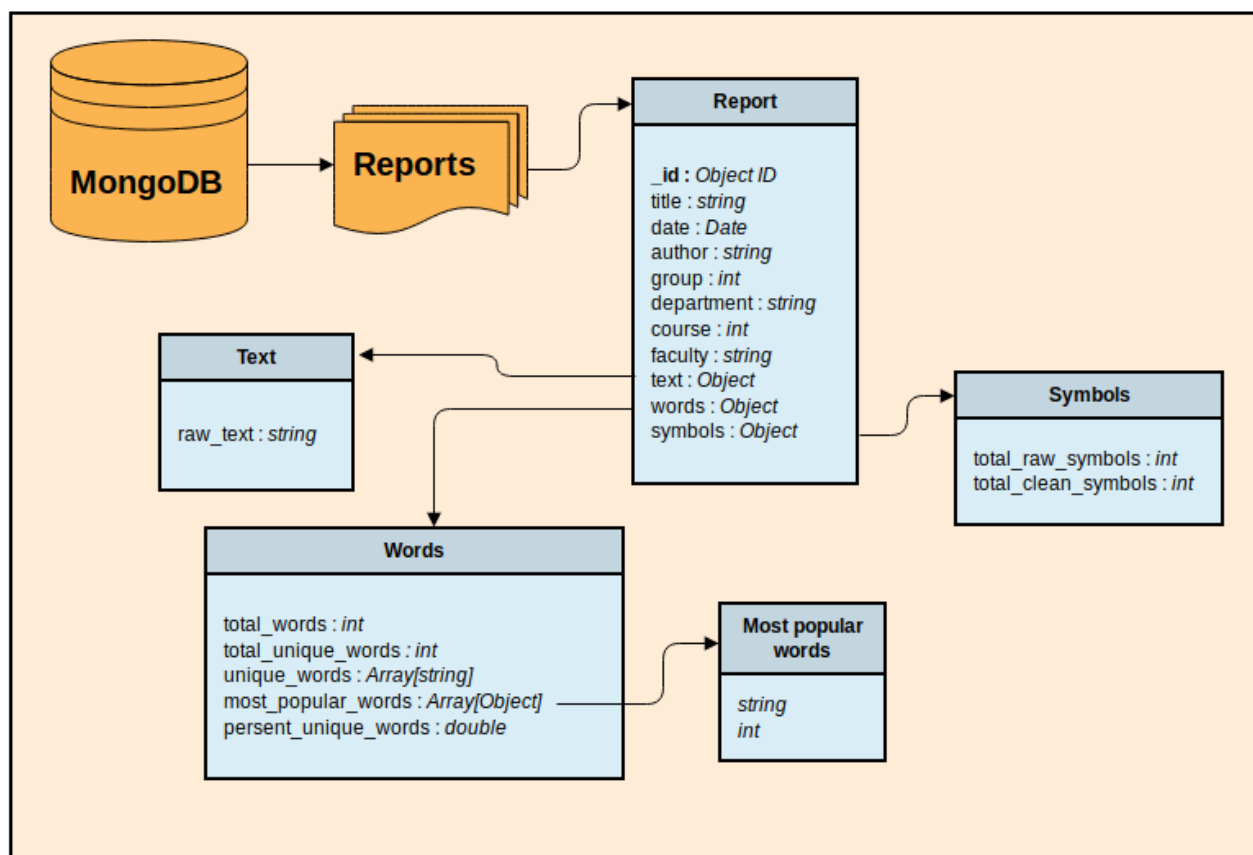
Пользователь выполняет опциональные шаги

Опциональные шаги

- Для просмотра статистики по студентам группы *пользователь* нажимает на **строчку** с интересующей его группой
- Для сравнения словарных запасов студентов *пользователь* на странице со статистикой группы выбирает нужных студентов с помощью **чек-боксов** и нажимает кнопку **“Сравнить”**
- Для просмотра подробной статистики и отчетов студента *пользователь* на странице со статистикой группы нажимает на **строчку** с интересующим его студентом
- Для редактирования полей отчета и просмотра подробной статистики по отчету *пользователь* на странице со статистикой студента нажимает на **строчку** с интересующим его отчетом

4. МОДЕЛЬ ДАННЫХ

NoSQL модель данных (MongoDB)



Подробное описание и расчёт объёма

В качестве СУБД - **MongoDB**. В БД одна коллекция - *Reports*, в которой содержатся отчёты. Документы (отчёты) имеют следующие поля:

****_id**** - идентификатор документа. Имеет тип - *ObjectID*. **V = 12b**

title - название отчёта. Имеет тип - *string*. $V = 2b * Nt$, где Nt - средняя длина названия отчёта, пусть $Nt = 20$. Тогда **V = 40b**.

date - дата последней модификации отчёта. Имеет тип - *date*. **V = 8b**

author - ФИО автора отчёта. Имеет тип - *string*. $V = 2b * Na$, где Na - средняя длина ФИО автора отчёта, пусть $Na = 25$. Тогда **V = 50b**.

group - номер группы автора отчёта. Имеет тип - *int*. **V = 4b**

department - кафедра автора отчёта. Имеет тип - *string*. $V = 2b * Nd$, где Nd - средняя длина названия кафедры автора отчёта, пусть $Nd = 10$. Тогда **V = 20b**.

course - номер курса автора отчёта. Имеет тип - *int*. **V = 4b**

faculty - факультет автора отчёта. Имеет тип - *string*. $V = 2b * Nf$, где Nf - средняя длина названия факультета автора отчёта, пусть $Nd = 10$. Тогда **V = 20b**.

text - вложенный документ, который содержит информацию о тексте отчёта. Имеет тип - *Object*

raw_text - текст отчёта. Имеет тип - *string*. $V = 2b * Nrt$, где Nrt - среднее количество символов в отчёте, пусть $Nrt = 5000$. Тогда **V = 10000b**.

words - вложенный документ, который содержит информацию о словах отчёта. Имеет тип - *Object*

total_words - всего слов в отчёте. Имеет тип - *int*. **V = 4b**

total_unique_words - всего уникальных слов в отчёте. Имеет тип - *int*. **V = 4b**

unique_words - массив уникальных слов в отчёте. Имеет тип - *Array[string]*. $V = 2b * Nw * Nuw$, где Nw - средняя длина слова в отчёте, а Nuw - среднее количество уникальных слов в отчёте, пусть $Nw = 10$; $Nuw = 100$. Тогда **V = 2000b**.

most_popular_words - массив пар слово-частота самых частых слов в отчёте. Имеет тип - *Array[(string, int)]*. $V = (2b * Nw + 4b) * 20$. **V = 480b**

percent_unique_words - процент уникальных слов в отчёте. Имеет тип - *double*. **V = 8b**

symbols - вложенный документ, который содержит информацию о символах отчёта. Имеет тип - *Object*

total_raw_symbols - количество символов в исходном тексте отчёта. Имеет тип - *int*. **V = 4b**

total_clean_symbols - количество символов в очищенном тексте отчёта. Имеет тип - *int*. **V = 4b**

Итого объем документа в среднем **12662b = 12.4kb**. Объем всей БД **V = 12.4kb * N**, где N количество отчётов в БД.

Например, если $N = 1000$, то объём БД 12.1mB

Запросы

Пример запроса на добавление отчёта

```
db['reports'].insert_one({
  'title': 'Курсовая',
  'date': '01.01.2012',
  'author': 'Иванов И.И.',
  'group': 1111,
  'department': 'МОЭВМ'
  'course': 4
  'faculty': 'КТИ'
  'text': {
    'raw_text': 'отчёт и курсовая если .... NoSQL'
  }
  'words': {
    'total_words': 10,
    'total_unique_words': 4,
    'unique_words': [
      'отчёт', 'курсовая', 'БД', 'NoSQL'
    ]
    'most_popular_words': [
      ('отчёт': 3), ('курсовая': 5)
    ]
  },
  ],
```

```

        'percent_unique_words': 40.0
    }
    'symbols': {
        'total_raw_symbols': 100,
        'total_clean_symbols': 50
    }
}
}))

```

Для добавления отчёта в БД нужен только один запрос. Например, чтобы добавить 1000 отчётов, запросов понадобится 1000

Пример запроса на получение всех отчётов автора

```
db['reports'].find({'author': 'Иванов И.И.'}).sort('title')
```

Для получения всех отчетов автора нужен только один запрос. Также, чтобы ускорить выполнения данного запроса добавлен индекс по полю *author*:

```
db['reports'].create_index('author')
```

Пример запроса на получение статистики по группе

```

db['reports'].aggregate([
    {'$match': {'group': group}},
    {'$group': {
        '_id': '$author',
        'avg_total_words': {'$avg': '$words.total_words'},
        'avg_unique_words': {'$avg': '$words.total_unique_words'},
        'avg_percent_unique_words': {'$avg': '$words.percent_unique_words'}
    },
    {
        'unique_words': {'$addToSet': '$words.unique_words'},
        'avg_total_raw_symbols': {'$avg': '$symbols.total_raw_symbols'},
        'avg_total_clean_symbols': {'$avg': '$symbols.total_clean_symbols'}
    },
    {
        'total_reports_loaded': {'$sum': 1}
    }
    ],
    {'$addFields': {
        'unique_words': {
            '$reduce': {
                'input': '$unique_words',
                'initialValue': [],
                'in': {'$setUnion': ['$value', '$$this']}
            }
        }
    }},
    {'$addFields': {'total_unique_words': {'$size': '$unique_words'}}},
    {'$sort': {'_id': 1, 'total_unique_words': -1}}
])

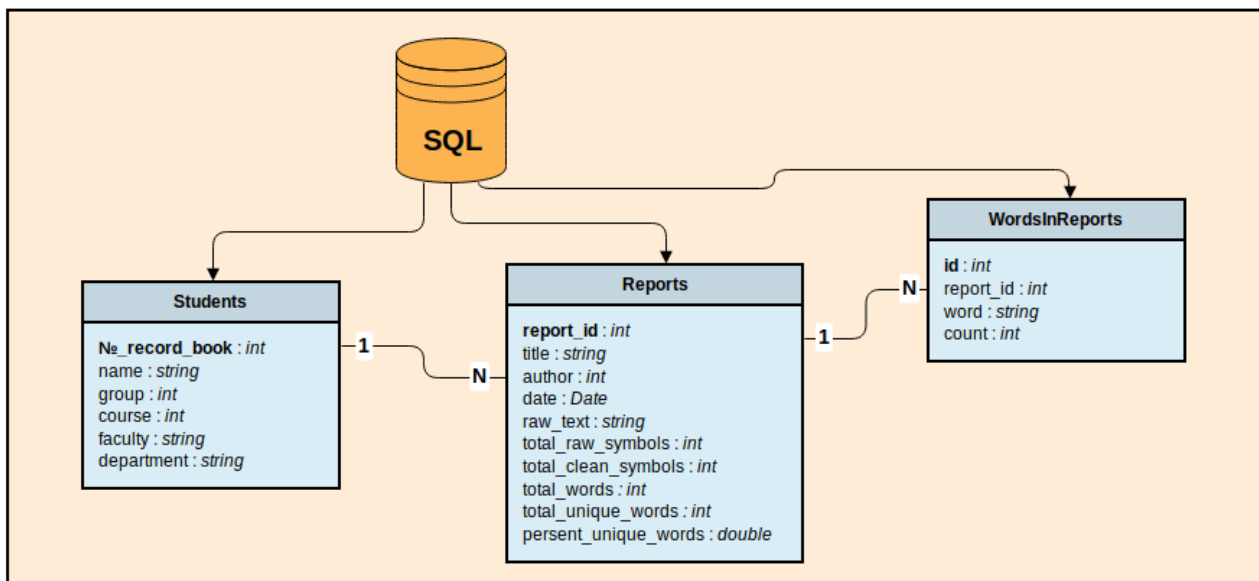
```

Для получения статистики по группе нужен только один запрос. Также для ускорения в БД добавлены индексы по полям *author* и *group*:

```

self.db['reports'].create_index('group')
self.db['reports'].create_index('author')

```



Подробное описание и расчёт объёма

В БД будут три таблицы:

- Students* - таблица, содержащая информацию о студентах, которые загружали отчёты в БД. Включает поля:

****№_record_book**** - № зачётной книжки. Имеет тип - *int*. **V = 4b**

name - ФИО студента. Имеет тип - *string*. $V = 2b * Na$, где Na - средняя длина ФИО, $Na = 25$ из пред. пункта. Тогда **V = 50b**.

group - номер группы. Имеет тип - *int*. **V = 4b**

course - номер курса. Имеет тип - *int*. **V = 4b**

faculty - факультет. Имеет тип - *string*. $V = 2b * Nf$, где Nf - средняя длина названия факультета, $Nd = 10$ из пред. пункта. Тогда **V = 20b**.

department - кафедра. Имеет тип - *string*. $V = 2b * Nd$, где Nd - средняя длина названия кафедры, $Nd = 10$ из пред. пункта. Тогда **V = 20b**.

Итого на одну запись в таблице **102b** или **0.01kB**

- Reports* - таблица, содержащая информацию об отчётах, загруженных в БД. Включает поля:

report_id - уникальный номер отчёта. Имеет тип - *int*. **V = 4b**

title - название отчёта. Имеет тип - *string*. $V = 2b * Nt$, где Nt - средняя длина названия отчёта, $Nt = 20$ из пред. пункта. Тогда **V = 40b**.

author - внешний ключ к таблице *Students*; указывает на автора отчёта. Имеет тип - *int*. **V = 4b**

date - дата последней модификации отчёта. Имеет тип - *date*. **V = 8b**

raw_text - текст отчёта. Имеет тип - *string*. $V = 2b * Nrt$, где *Nrt* - среднее количество символов в отчёте, *Nrt* = 5000 из пред. пункта. Тогда **V = 10000b**.

total_raw_symbols - количество символов в исходном тексте отчёта. Имеет тип - *int*. **V = 4b**

total_clean_symbols - количество символов в очищенном тексте отчёта. Имеет тип - *int*. **V = 4b**

total_words - всего слов в отчёте. Имеет тип - *int*. **V = 4b**

total_unique_words - всего уникальных слов в отчёте. Имеет тип - *int*. **V = 4b**

percent_unique_words - процент уникальных слов в отчёте. Имеет тип - *double*. **V = 8b**

Итого на одну запись в таблице **10080b** или **9.85kB**

- *WordsInReports* - таблица, содержащая информацию о уникальных словах и их количестве в отчёте. Включает поля:

id - уникальный № записи. Имеет тип - *int*. **V = 4b**

report_id - внешний ключ к таблице *Reports* - информация, в каком отчёте встретилось слово. Имеет тип *int*. **V = 4b**

word - слово. Имеет тип *string*. $V = 2b * Nw$, где *Nw* - средняя длина слова в отчёте, *Nw* = 10 из пред. пункта. **V = 20b**

count - количество повторений в отчёте. Имеет тип *int*. **V = 4b**

Итого на одну запись в таблице **32b** или **0.03kB**

Пусть:

Nlr - среднее количество отчётов, которое загрузил каждый студент. *Nlr* = 5

Nuw - среднее количество уникальных слов в отчёте. *Nuw* = 100 из пред. пункта

Итого для хранения *N* отчётов в бд понадобится: $V = 102b * N / Nlr + 10088b * N + 32b * Nuw * N = N * (20.4b + 10088b + 32b * 100) = N * 13308.4b = N * 13kB$.

Например, если $N = 1000$ (как в пред. пункте), то объём БД 98.5mB

Запросы

Пример запроса на добавление студента

```
INSERT INTO Students (1111, 'Иванов И.И.', 6304, 3, 'КТИ', 'МОЭВМ')
```

Пример запросов, которые требуются для добавления отчёта

```
INSERT INTO Reports (111212, 'Курсовая', 1111, '01.01.2018', 'Курсовая работа и  
исследование по База данных вывода задача технологии работа ЛЭТИ', 100, 50, 15,  
5, 0.33)
```

```
INSERT INTO WordsInReports (3402340, 111212, 'работа', 2)
```

```
INSERT INTO WordsInReports (3402341, 111212, ..., ...)
```

```
INSERT INTO WordsInReports (3402342, 111212, ..., ...)
```

...

Пусть:

Nlr - среднее количество отчётов, которое загрузил каждый студент. $Nlr = 5$

Nuw - среднее количество уникальных слов в отчёте. $Nuw = 100$ из пред. пункта

Итого, чтобы добавить N отчётов в БД потребуется $N / Nlr + N + N * Nuw = N * (0.2 + 1 + 100) = N * 101.2$ запросов.

Например, если $N = 1000$ (как в пред. пункте), то запросов 101200

Пример запроса на получение всех отчётов автора

```
SELECT * FROM Reports
JOIN Students ON author=№_record_book
WHERE name='Иванов И.И.'
```

SQL vs NoSQL

SQL модель данных занимает больше места, т.к. SQL нет поддержки массивов и слова отчёта потребуется хранить как записи в отдельной таблице, что добавляет значительные накладные расходы за счёт дублирования информации

По сравнению с SQL в NoSQL модели данных есть дублирование информации о студенте, загрузившем отчёт, но по сравнению, с местом которое требуется для хранения текста отчёта и слов, входящих в отчёт, это дублирование не вносит практически никакого вклада

В SQL для загрузки N отчётов в БД потребуется $N * 101.2$ запросов, а в NoSQL **один** запрос

В итоге в SQL большее дублирование информации и расход места чем в NoSQL. К тому же *MongoDB* является документоориентированной СУБД и ее прямое назначение - работа со слабосвязанными и слабоструктурированными данными. Так что выбор очевиден в пользу **MongoDB**

5. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

Приложение позволяет загружать студенческие отчёты в формате docx и получать агрегированную статистику по ним для групп (с фильтром по факультету, кафедре и курсу), расширенную для отдельной группы, а также для отдельного студента. Предусмотрена возможность сравнивать словарные запасы (рассчитанные по всем отчётам студента) между одноклассниками.

Экраны приложения



Рисунок 1 – Главная страница

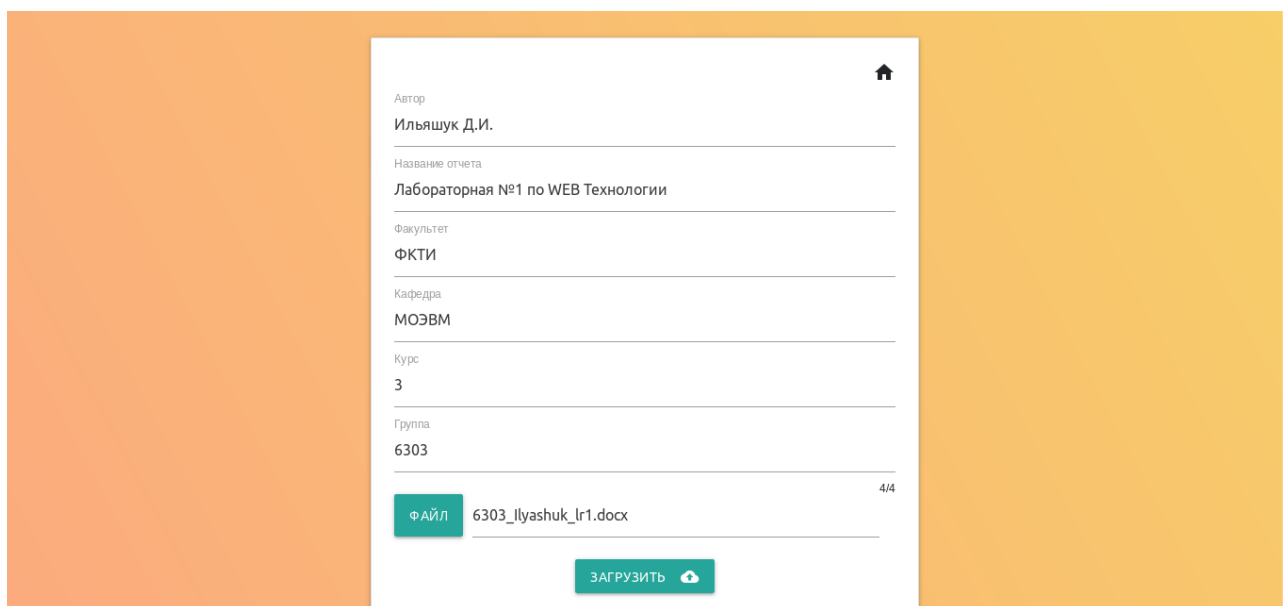
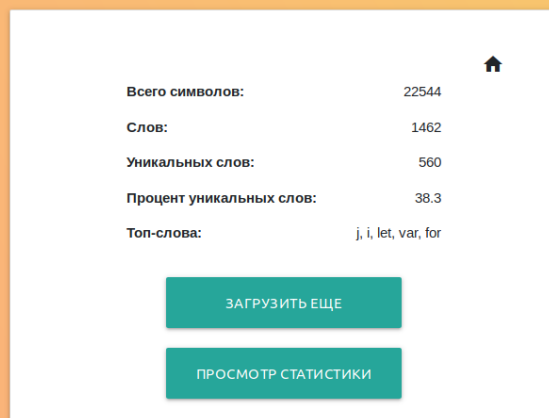


Рисунок 2 – Загрузка отчёта



Всего символов: 22544

Слов: 1462

Уникальных слов: 560

Процент уникальных слов: 38.3

Топ-слова: j, i, let, var, for

ЗАГРУЗИТЬ ЕЩЕ

ПРОСМОТР СТАТИСТИКИ

Рисунок 3 – Статистика после загрузки отчёта



Статистика по группам

Факультет: Любой

Кафедра: Любой

Курс: Любой

#	Группа	Средний процент уникальных слов	Всего слов в среднем	Уникальных слов в среднем	Отчетов загружено
1	6303	40	610	237	3
2	6304	46	797	212	7
3	6381	45	354	144	3

Рисунок 4 – Статистика по группам

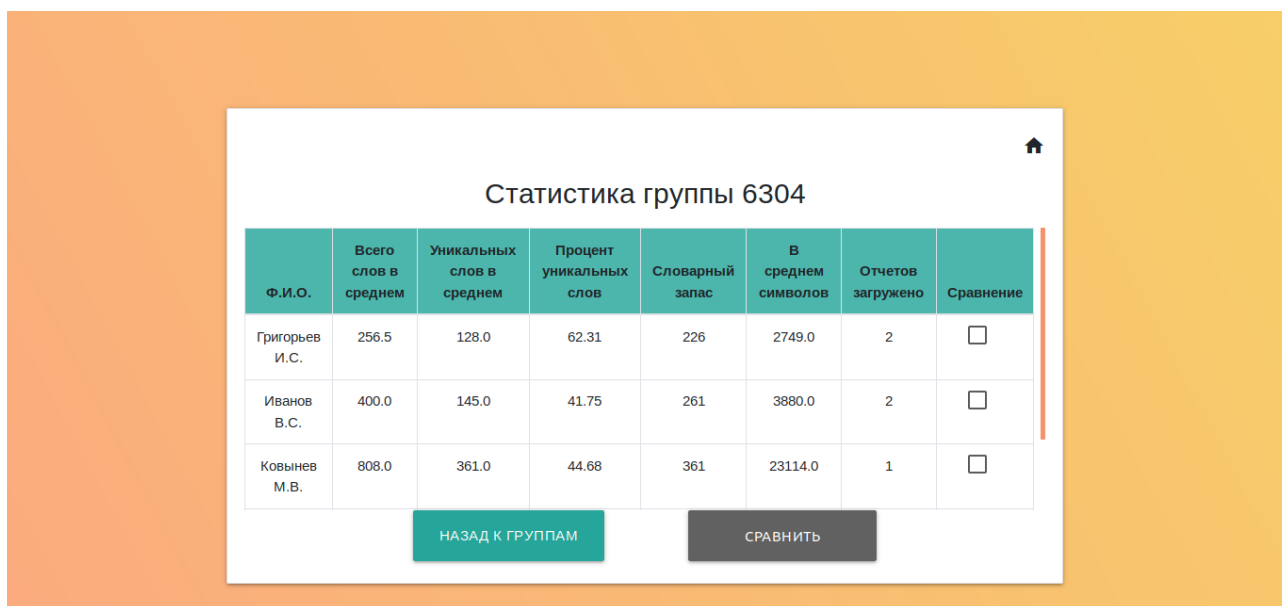


Рисунок 5 – Статистика по группе

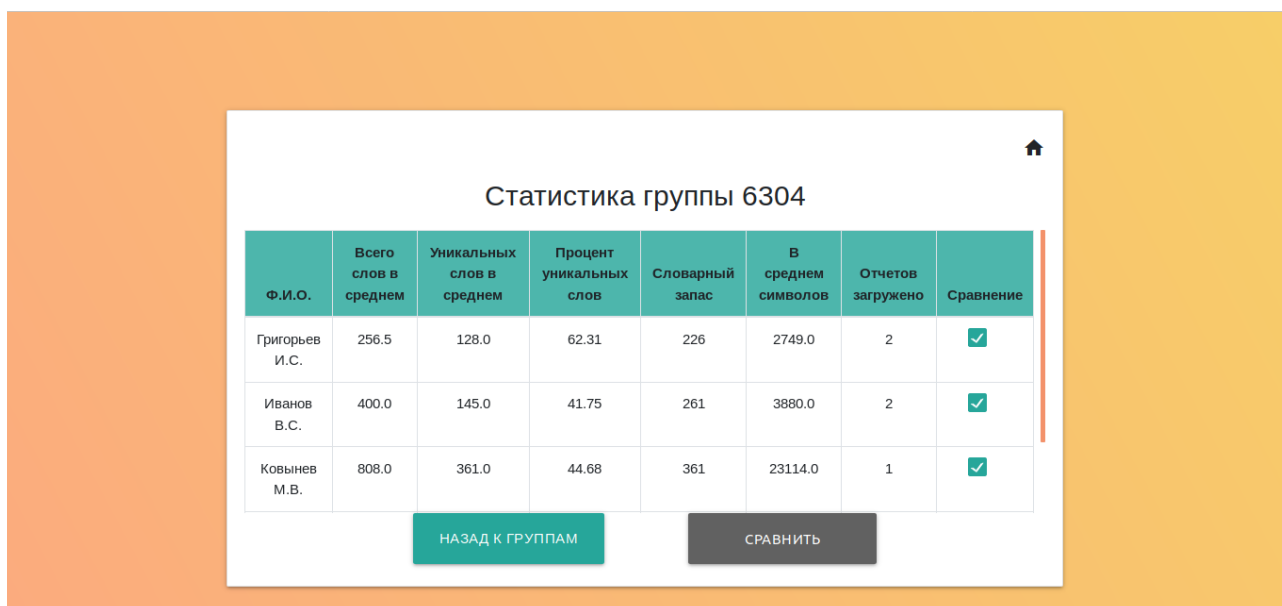


Рисунок 6 – Выбраны студенты для сравнения

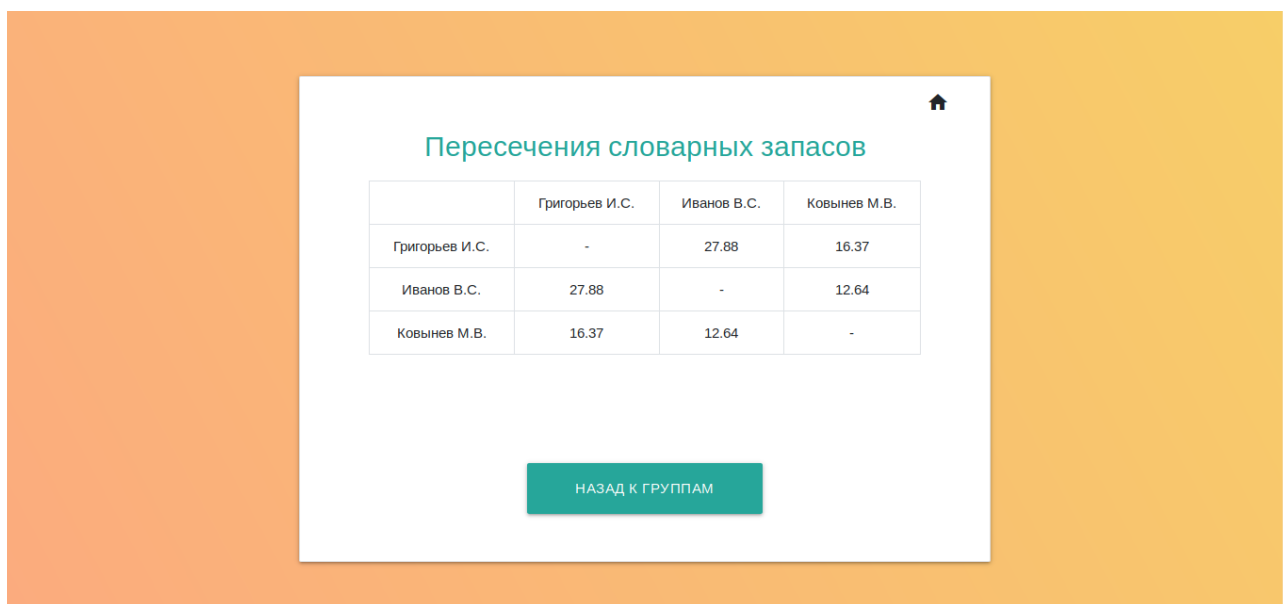


Рисунок 7 – Сравнение словарных запасов студентов

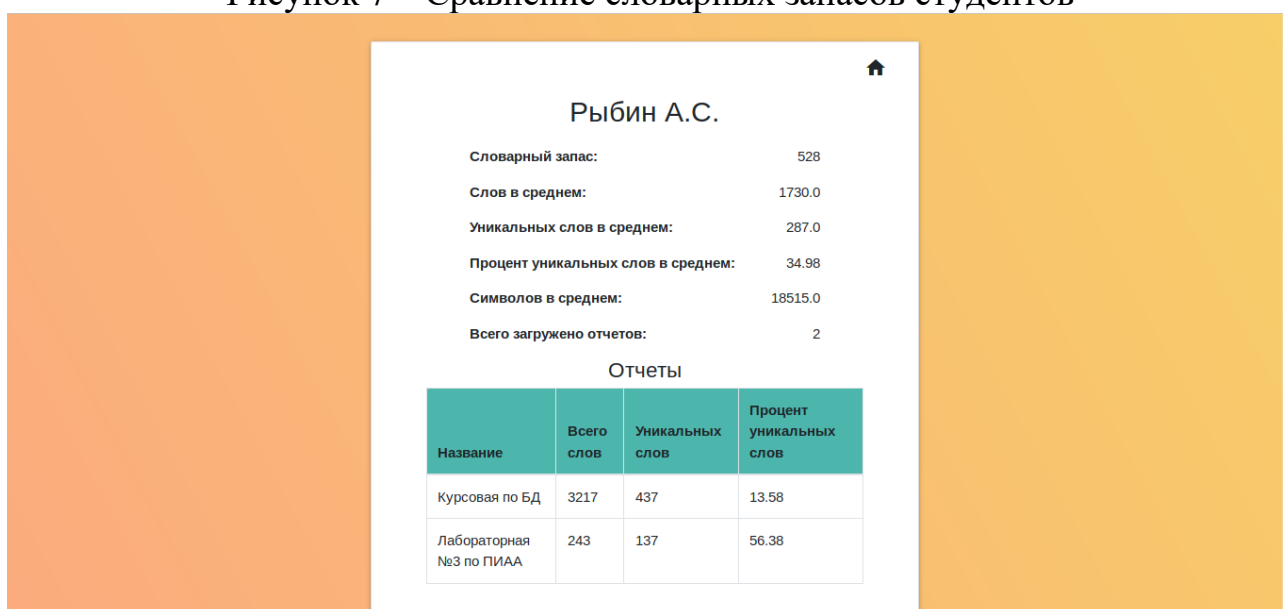


Рисунок 8 – Статистика по студенту

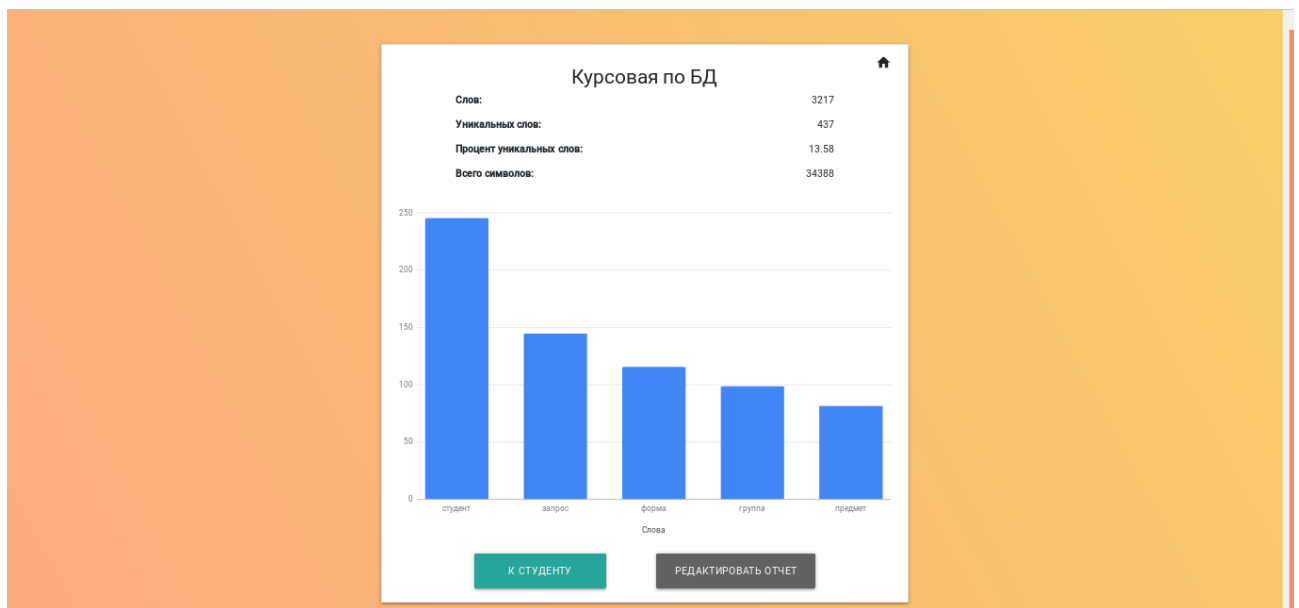


Рисунок 9 – Статистика по отчёту

Использованные технологии

Back-end представляет из себя *python* приложение с использованием фреймворка *Flask*, а для хранения данных задействована *MongoDB*. Приложение предоставляет *REST API* для загрузки отчётов и получения статистики по ним.

Front-end реализован с помощью встроенного во *Flask* шаблонизатора *Jinja2*, *JS* библиотеки *Jquery* и *CSS* фреймворков *Bootstrap*, *W3*.

Ссылки:

- [Репозиторий проекта](#)
- [Wiki проекта](#)

ВЫВОДЫ

В итоге выполнения ИДЗ реализовано приложение, позволяющее загружать, систематизировано хранить и обрабатывать студенческие отчёты.

Однако приложение имеет следующие недостатки:

1. Ограниченность формата отчёта (только **docx**)
2. Хранение данных о студентах косвенно, в коллекции предназначенной для хранения отчётов. Из этого следует: отсутствие авторизации, возможны дубликаты отчётов одного и того же студента, противоречивая информация об отчёте и т.д.
3. Скучный и малофункциональный Веб-интерфейс

Возможные пути улучшения:

1. Добавить поддержку pdf, odt, plain text
2. Реализовать более полную модель данных. Включить информацию о студентах в отдельную коллекцию, информацию о структуре кафедр и факультетов также включить в отдельную коллекцию.
3. Модифицировать веб-интерфейс с использованием современных JS фреймворков (React, Angular, Vue)

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. NoSQL // Википедия. URL: <https://ru.wikipedia.org/wiki/NoSQL> (дата обращения: 19.04.2019).
2. MongoDB Docs // WEB сайт MongoDB. URL: <https://docs.mongodb.com/> (дата обращения: 19.04.2019).
3. Welcome to Flask // WEB сайт Flask. URL: <http://flask.pocoo.org/docs/1.0/> (дата обращения: 19.04.2019).
4. jQuery API // WEB сайт JQuery. URL: <https://api.jquery.com/> (дата обращения: 19.04.2019).
5. Bootstrap // WEB сайт Bootstrap. URL: <https://getbootstrap.com/> (дата обращения: 19.04.2019).
6. Python 3.7.3 documentation// WEB сайт Python. URL: <https://docs.python.org/3/> (дата обращения: 19.04.2019).

ПРИЛОЖЕНИЕ А

ИНСТРУКЦИЯ ПО СБОРКЕ И ЗАПУСКУ

1. Склонировать репозиторий проекта
2. Перейти в папку с репозиторием
3. Собрать *docker* образ командой

```
docker-compose build
```

4. Создать *volume*, в котором персистентно будут храниться данные БД с помощью команды

```
docker volume create nosql1h19-report-stats-data
```

5. Запустить контейнер командой

```
docker-compose up -d
```

6. Перейти на **localhost:5000**

7. Для остановки контейнера выполнить команду

```
docker-compose stop
```